Web aplikacija za chat

Čuković, Josip

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

WEB APLIKACIJA ZA CHAT

Završni rad

Josip Čuković

Osijek, 2020.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac Z1P - Obrazac za ocjenu završnog rada na preddiplomskom sveučilišnom studiju

Osijek, 17.09.2020.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada na preddiplomskom sveučilišnom studiju

Ime i prezime studenta:	Josip Čuković
Studij, smjer:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R4046, 19.09.2019.
OIB studenta:	56153278641
Mentor:	Izv. prof. dr. sc. Krešimir Nenadić
Sumentor:	
Sumentor iz tvrtke:	
Naslov završnog rada:	Web aplikacija za chat
Znanstvena grana rada:	**Informacijski sustavi (zn. polje računarstvo)**
Predložena ocjena završnog rada:	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 2 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 2 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene mentora:	17.09.2020.
Datum potvrde ocjene Odbora:	23.09.2020.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 24.09.2020.

Ime i prezime studenta:	Josip Čuković
Studij:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R4046, 19.09.2019.
Turnitin podudaranje [%]:	10%

Ovom izjavom izjavljujem da je rad pod nazivom**: Web aplikacija za chat**

izrađen pod vodstvom mentora Izv. prof. dr. sc. Krešimir Nenadić

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SADRŽAJ

1. UVOD

Komunikacija putem interneta postala je neizostavan dio ljudske svakodnevnice. Sve užurbaniji način života zahtjeva jednostavnu i brzu komunikaciju te lakšu povezanost među ljudima. Sve veći društveno-tehnološki napredak ispunjava zahtjeve današnjeg društva za razmjenom informacija jednostavnim „klikom miša“.

Aplikacija za chat, odnosno razmjenu poruka, čija će izrada biti opisana i objašnjena u nastavku ovoga rada, pruža jednostavne, ali ključne funkcionalnosti koje učinkovito ispunjavaju svrhu same aplikacije, a to je komunikacija i povezivanje njezinih korisnika.

Kroz poglavlja je opisano sve što je potrebno za rad aplikacije i prikazano je kako su pojedini elementi spojeni u jedinstvenu i funkcionalnu cjelinu. Potrebno je izraditi web aplikaciju kojom se korisnicima omogućava nesmetana komunikacija porukama u stvarnom vremenu. Nakon uvoda, u drugom poglavlju, opisane su tehnologije korištene pri izradi aplikacije. Za svaku tehnologiju su objašnjene njezine najvažnije karakteristike. U trećem poglavlju, opisano je kako su pojedini dijelovi programa spojeni u cjelinu, izrada aplikacije, odnosno način na koji svaki specifični dio aplikacije zapravo funkcionira. Aplikacija je objašnjavana na način kojim se sam korisnik kreće kroz aplikaciju, uključujući prijavu i registraciju, potvrđivanja email adrese, slanja poruka pa sve do promjene slike profila i pretrage korisnika po imenu. U četvrtom poglavlju prikazan je izgled same aplikacije pri određenim stanjima odnosno svih njezinih dijelova. U zadnjem poglavlju ukratko su rečene prednosti koje pruža sama aplikacija kao i sve tehnologije korištene prilikom izrade ove web aplikacije.

1.1. Zadatak završnog rada

Zadatak ovog završnog rada je izrađivanje web aplikacije za chat, pri čemu je potrebno savladati i koristiti tehnologije i alate koji su navedeni kasnije u radu. Potrebno je kreirati aplikaciju koja prijavljenim i verificiranim korisnicima uz ostale funkcionalnosti dopušta razmjenu poruka i izmjenu slike vlastitog profila.

2. TEHNOLOGIJE KORIŠTENE U IZRADI APLIKACIJE

U nastavku su navedene i ukratko objašnjenje tehnologije koje su korištene pri izradi web aplikacije.

2.1. HTML

Hyper Text Markup Language, poznatiji kao HTML, standardni je opisni jezik za stvaranje HTML dokumenata, tj. web stranica. Korištenjem predefiniranih oznaka (engl. *tag*) opisana je struktura web stranice. Pomoću HTML koda preglednik zna kako prikazati sadržaj korisniku. Važno je napomenuti da preglednik ne prikazuje oznake nego ih koristi kako bi prikazao sadržaj unutar pojedinog elementa. HTML oznake su imena elemenata unutar kutnih zagrada i imaju sljedeći oblik: <ime_oznake>…Sadržaj….</ime_oznake> [1]. Elemente je moguće ugnježđivati jedne u druge, ali pritom treba biti vrlo oprezan kod zatvaranja oznaka jer se unutarnja oznaka mora zatvoriti prije vanjske oznake.

```
1  <!DOCTYPE html>
2  <html lang="en">
3    <head>
4      <meta charset="UTF-8" />
5      <title>Document</title>
6    </head>
7    <body>  </body>
8  </html>
```

Slika 2.1. *Osnovna struktura HTML dokumenta*

Slikom 2.1. prikazana je osnovna struktura HTML dokumenta gdje oznaka <!DOCTYPE html> označava da se radi o HTML 5 dokumentu. Oznake <html> i </html> čine element koji obuhvaća sav sadržaj na stranici te se navodi kao korijenski element. Nadalje, oznake <head> i </head> sadržavaju meta podatke, tj. podatke o podatcima. Meta podatci definiraju ime dokumenta, CSS stilove za stiliziranje sadržaja, skripte i ostale meta podatke. Oznake <title> i </title> označavaju naslov dokumenta, dok element <meta charset="UTF-8"> postavlja znakovni skup koji će HTML dokument koristiti. Unutar oznaka <body> i </body> nalazi se sav vidljivi sadržaj web stranice.

2.2. CSS

CSS (engl. *Cascading Style Sheets*) je jezik kojim je opisan stil HTML dokumenta, odnosno prezentacija web stranica [2]. Opći oblik CSS deklaracije ima sljedeći oblik: selektor:{svojstvo: vrijednost} gdje selektor određuje element na koje se stilsko sredstvo odnosi. Neki najčešće korišteni selektori su: jednostavni selektor, klasni selektor, id selektor.

CSS se može primijeniti na HTML dokument na 3 načina : pomoću vanjske primjene liste stilova (engl. *external stylesheet*), unutarnje liste stilova (engl. *internal stylesheet*) te stil unutar linije (engl. *inline style*) [3]. Kako bi se koristila vanjska lista HTML dokument se mora povezati s CSS dokumentom. To se radi pomoću elementa <link> unutar <head> elementa. Slikom 2.2 prikazan je primjer vanjske primjene liste stilova. Ovaj način uporabe CSS datoteka zapravo je najpoželjniji zato što se ista datoteka može primijeniti na više HTML dokumenata.

```
1  <!DOCTYPE html>
2  <html lang="en">
3    <head>
4      <meta charset="UTF-8" />
5      <title>Document</title>
6      <link rel="stylesheet" href="index.css" />
7    </head>
8    <body>
9      <p>Text</p>
10   </body>
11 </html>
```

Slika 2.2. *Primjer vanjske primjene liste sstilova*

```
# index.css > ...
1   p {
2     color: red;
3   }
```

Slika 2.3. *Primjer promjene boje teksta*

Slikom 2.3. prikazan je sadržaj CSS datoteke korištene u primjeru prikazanom slikom 2.2.. Unutarnja lista stilova definira se unutar <style> elementa unutar <head></head> oznaka. Ovakva primjena CSS-a može dovesti do dupliciranja koda zato što CSS primijenjen unutarnjom listom stilova mora biti ponovno napisan unutar svakog HTML dokumenta. Primjer unutarnje liste stilova prikazan je slikom 2.3.

```
<head>
  <meta charset="UTF-8" />
  <title>Document</title>
  <style>
    p {
      color: red;
    }
  </style>
</head>
```

Slika 2.3. *Primjer unutarnje liste stilova*

Stil unutar linije predstavlja CSS deklaraciju koja utječe na samo jedan element, a nalazi se unutar style atributa. Ovakav način stiliziranja stranice preporučljivo je izbjegavati, osim ako to nije nužno, zato što se miješa sadržaj s prezentacijom, kod postaje težak za čitanje, održavanje i slično. Slikom 2.4. prikazan je primjer stila unutar linije.

```
<body>
  <p style="color: red;">Text</p>
</body>
```

Slika 2.4. *Primjer stila unutar linije*

2.3. Materialize

Materialize je responzivni CSS okvir (engl. *framework*) kojeg je Google kreirao i dizajnirao [4]. Baziran je na konceptima Google-ova materijalnog dizajna te čini stvaranje makete web stranica vrlo jednostavnim. Neke komponente koje materialize nudi su: gumbovi (engl. *buttons*), navigacijska traka (engl. *navigation bar*), unosne forme (engl. *form inputs*), boje i sjene, izgled rešetki (engl. *grid layout*) i mnoge druge [5]. Materialize također pruža i interaktivne komponente kao što su skočni prozori (engl. *modals/pop-ups*), kartice (engl. *tabs*), padajući izbornici (engl. *dropdown*) i mnogo drugih. Kako bi Materialize bio primijenjen na određeni element, jednostavno se mora u ime klase dodati ključnu riječ.

```
<body>
  <p class="red-text blue">Text</p>
</body>
```

Slika 2.5. *Primjer korištenja Materialize-a*

Slikom 2.5. prikazan je primjer kojim je promijenjena boja teksta u crvenu, a pozadina u plavu boju.

2.4. JavaScript

JavaScript je skriptni programski jezik koji se izvršava u web pregledniku na strani korisnika [6]. Ovaj programski jezik je jezik visoke razine (engl. *high-level*). Uz HTML i CSS, JavaScript je jedna od bitnijih tehnologija za „World Wide Web“. JavaScript omogućava interaktivne web stranice i ključan je dio web aplikacija. Velika većina web stranica koristi ovaj programski jezik za ponašanje stranice na strani korisnika/klijenta. Kao više paradigmatski (engl. *multi-paradigm*) jezik, JavaScript podržava programiranje pogonjeno događajima (engl. *event-driven*), funkcijske i imperativne programske stilove [7]. Kako bi koristili JavaScript funkcionalnosti unutar HTML dokumenta moramo ga uključiti pomoću <script> unutar <head> dijela HTML dokumenta, unutar <body> dijela HTML dokumenta ili pak unutar oba dijela. Kao što je prikazano kao slikom 2.6., JavaScript se može koristiti kao odvojena datoteka ili može biti napisana unutar HTML dokumenta. Pomoću ovog programskog jezika može se vrlo jednostavno utjecati na HTML elemente. Najučestalije korištene metode su getElementById() i getElementsByClassName(). Slikom 2.7. prikazan je primjer korištenja prve navedene metode kojom se u unutrašnjost HTML elementa upisuje željeni string.

```
  <script>
    document.getElementById("ID_ELEMENTA").style.display = "none";
  </script>
</body>
```

Slika 2.6. *Primjer korištenja JavaScript programskog jezika unutar <body> elementa*

```
document.getElementById("ID_elementa").innerHTML ="Pozdrav";
```

Slika 2.7. *Korištenje getElementById() metode*

JavaScript programskim jezikom moguće je mijenjati atribute HTML elemenata, CSS stilove (primjerice veličina fonta), sakriti ili pokazati određeni element (ako se uvjeti ispunjavaju) i slično.

```
document.getElementById("ID_ELEMENTA").style.fontSize = "16px";
```

Slika 2.8. *Promjena fonta korištenjem JavaScript programskog jezika*

```
document.getElementById("ID_ELEMENTA").style.display = "none";
```

Slika 2.9. *Sakrivanje elementa korištenjem JavaScript programskog jezika*

2.5. Firebase

Firebase je razvila tvrtka Firebase Inc. 2011. godine te predstavlja platformu za razvoj mobilnih i web aplikacija [8]. Navedenu platformu Google je otkupio 2014. godine[8]. Firebase platformom je pružen veliki dio usluga koje bi programeri inače morali sami kreirati. Neke usluge su: Firebase Realtime Database, Firebase Authentication, Firebase Storage, Firebase Hosting i mnoge druge. Firebase Realtime database predstavlja bazu podataka u stvarnom vremenu.

Firebase Realtime database omogućuje pohranu i sinkronizaciju podataka u stvarnom vremenu što otvara mnoge mogućnosti. Firebase Authentication je usluga koja omogućuje provjeru autentičnosti korisnika putem e-maila i lozinke, Google računa, broja mobitela i putem brojnih drugih načina te razne mogućnosti koje ovise o statusu korisnika, tj. je li korisnik prijavljen u sustav ili ne. Firebase Storage omogućuje spremanje slika, zvuka, videa i ostalih sadržaja. Kako bi sve navedene usluge mogle biti korištene potrebno je prijaviti se pomoću Google računa na službenu Firebase stranicu.

3. IZRADA APLIKACIJE

Postupak izrade web aplikacije biti će opisan korak po korak u nastavku završnoga rada.

3.1. Prijava i registracija korisnika

Prije upotrebe web aplikacije korisnik se mora prijaviti, odnosno registrirati. To će napraviti jednostavnim klikom na *Login*, odnosno *Sign up* opciju. Nakon odabiranja opcije prikazati će se skočni prozor te je potrebno unijeti nekoliko informacija. Prilikom prijavljivanja korisnika će se tražiti unos odgovarajuće email adrese te zaporka. Slikama 3.1., 3.2. i 3.3 prikazan je HTML kod skočnih prozora. Unutar skočnog prozora za registraciju nalazi se registracijska forma ostvarena <form></form> HTML oznakama koja se sastoji od nekoliko HTML <input> oznaka koje služe za unošenje email adrese, lozinke, imena i prezimena te se na dnu forme nalazi *Sign up* tipka. Unutar skočnog prozora za prijavu nalazi se forma za prijavu korisnika koja je ostvarena HTML oznakama <form</form> te se unutar forme nalaze HTML oznake <input/> i Login tipka za podnošenje zahtjeva. Unutar <input/> oznaka u formi za prijavu unosimo email i zaporku te ukoliko se jedno od navedenih polja ostavi prazno ili se ne unesu odgovarajući podatci, prijava nije moguća.

```
  <li class="logged-out">
    <a href="#" class="modal-trigger" data-target="modal-login">Login</a>
  </li>
  <li class="logged-out">
    <a href="#" class="modal-trigger " data-target="modal-signup">Sign up</a>
  </li>
</ul>
```

Slika 3.1. *Prikaz Login-a i Sign up-a te pozivanje odgovarajućih skočnih prozora*

```
<div id="modal-signup" class="modal">
  <div class="modal-content">
    <h4>Sign up</h4>
    <br />
    <form id="signup-form">
      <input type="email" id="signup-email" placeholder="Email adress" required />
      <input type="password" id="signup-password" placeholder="Password" required />
      <input type="text" id="signup-name" placeholder="First name" required />
      <input type="text" id="signup-surname" placeholder="Last name" required />
      <button class="btn">Sign up</button>
    </form>
  </div>
</div>
```

Slika 3.2. *Izvedba skočnog prozora za registraciju*

```html
<div id="modal-login" class="modal">
  <div class="modal-content">
    <h4>Login</h4>
    <br />
    <form id="login-form">
      <input type="email" id="login-email" placeholder="Email address" required />
      <input type="password" id="login-password" placeholder="Password" required />
      <button class="btn">Login</button>
    </form>
  </div>
</div>
```

Slika 3.3. *Izvedba skočnog prozora za prijavu*

Za uređivanje, korišten je CSS i u slučaju skočnih prozora i Materialize. Slikom 3.4. prikazano je uključivanje CSS dokumenta u HTML dokument.

```html
<link rel="stylesheet" href="styles/index.css">
```

Slika 3.4. *Uključivanje CSS dokumenta*

Kada korisnik unese podatke te prilikom podnošenja (engl. *submit*) istih, slušatelj (engl. *listener*) koji je postavljen na skočni prozor dohvaća podatke iz pojedinih polja i poziva Firebase funkciju "*createUserWithEmailAndPassword*".

Navedenoj funkciji predaju se email i lozinka (engl. *password*). Firebase kreira korisnika s predanim emailom i lozinkom te mu dodjeljuje jedinstveni identifikacijski niz znakova odnosno UID. UID je jedinstven svakom korisniku te se pomoću njega mogu vršiti razne manipulacije. Primjerice, to su provjera korisnikove aktivnosti, slanje poruka i slično. Nakon kreiranja korisnika potrebno je ažurirati njegove informacije, stoga se pomoću funkcije "*updateProfile*", kojoj je predano ime korisnika, koje je prethodno dohvaćeno iz polja prilikom registracije, ažurira korisničko ime. Slikom 3.5. prikazan je način dohvaćanja trenutnog korisnika nad kojim je pozvana prethodno navedena funkcija.

```javascript
var currentUser = firebase.auth().currentUser;
```

Slika 3.5. *Dohvaćanje trenutnog korisnika*

```
signup.addEventListener("submit", (e) => {
  e.preventDefault();
  const email = document.getElementById("signup-email").value;
  const password = signup["signup-password"].value;
  const name = signup["signup-name"].value;
  const surname = document.getElementById("signup-surname").value;
  const fullname = name + " " + surname;

  auth.createUserWithEmailAndPassword(email, password).then(() => {

    var currentUser = firebase.auth().currentUser;

    currentUser.updateProfile({ displayName: fullname }).catch(function (error) {
      console.log(error)
    });

    ShowAlertMessage(currentUser);
    AddUserToDatabase(currentUser);
  })
    .catch((error) => {
      window.alert(error.message);
    });
});
```

Slika 3.6. *Prikaz funkcija odgovornih za kreiranja korisnika*

Slikom 3.6. prikazane su razne funkcije koje su pozvane prilikom kreiranja svakog korisnika kao i dohvaćanje raznih vrijednosti.

Funkcijom "*ShowAlertMesage*", kojoj se predaje trenutni korisnik, u slučaju da korisnik nije potvrdio email prikazuje poruku koja obavještava korisnika kako je potrebno potvrditi email kako bi nesmetano mogao koristiti aplikaciju. Slikom 3.7. prikazana je navedena funkcija.

```
function ShowAlertMessage(currentUser) {

  if (currentUser.emailVerified == false) {
    window.alert("Please Verify your email in order to use our chat!");
  }
}
```

Slika 3.7. *Prikaz funkcije ShowAlertMessage*

Funkcija "*AddUserToDatabase*", kojoj se također predaje trenutni korisnik, dodaje sve bitne informacije u bazu podataka odnosno Firebase Database te dohvaća i postavlja zadanu (engl. *default*) sliku.

Prilikom svakog spremanja korisnika unutar baze podataka postavlja mu se vrijednost vlastitoga čvora na jednaku vrijednost kao i UID korisnika. Ovime se postiže lakše pristupanje podatcima pojedinog korisnika na način da se jednostavno traži vrijednost čvora koja je jednaka njegovom jedinstvenom identifikacijskom nizu znakova.

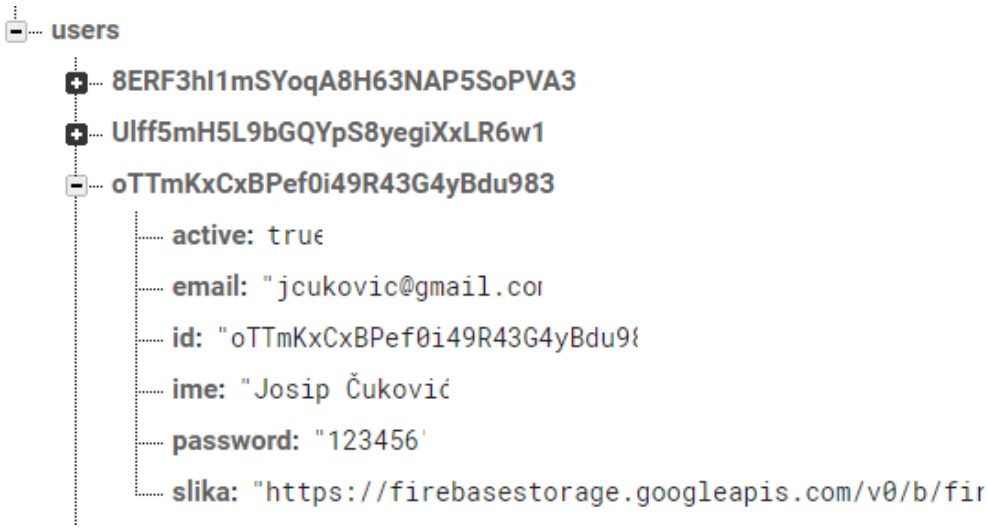
Slikom 3.8. prikazano je dodavanje podataka u bazu s određenom vrijednosti čvora. Slikom 3.9. prikazana je cijela funkcija koja je zadužena za strukturiranje i spremanje podataka u bazu podataka.

```
userRef.child(`${trenutniUser.uid}`).set(user);
```

Slika 3.8. *Dodavanje podataka u bazu podataka*

```
function AddUserToDatabase(trenutniUser) {
  const email = document.getElementById("signup-email").value;
  const password = signup["signup-password"].value;
  const name = signup["signup-name"].value;
  const surname = document.getElementById("signup-surname").value;
  const fullName = name + " " + surname;
  const modal = document.querySelector("#modal-signup");

  defaultPicture.getDownloadURL().then((link) => {

    const user = {
      slika: link,
      active: true,
      id: trenutniUser.uid,
      email,
      password,
      ime: fullName,
    };

    userRef.child(`${trenutniUser.uid}`).set(user);
    M.Modal.getInstance(modal).close();
    signup.reset();
    location.reload();
  });
}
```

Slika 3.9. *Prikaz funkcije AddUserToDatabase*



Slika 3.10. *Struktura podataka unutar baze podataka*

Kako bi korisnik potvrdio svoj email potrebno je kliknuti na tipku pod nazivom *Send Verification.* Prilikom klika poziva se funkcija "*SendEmail*" koja poziva funkciju pod nazivom "*sendEmailVerification*". Tada se korisniku prikazuje obavijest kako je email poslan na odgovarajuću email adresu. Slikom 3.11. prikazana je funkcija koja je zadužena za slanje emaila kako bi korisnik mogao potvrditi valjanost navedene email adrese.

```
function SendEmail() {
  var currentUser = firebase.auth().currentUser;

  currentUser.sendEmailVerification().then(function () {
    window.alert("Email has been sent, please log in again after verification.");
    auth.signOut();
    location.reload();
  }).catch(function (error) {

    window.alert(error.message)
  });
}
```

Slika 3.11. *Prikaz funkcije SendEmail*

Nakon što je korisnik stvorio svoj račun i odradio sve korake kako bi mogao koristiti aplikaciju može se prijaviti (engl. *login*) i koristiti ju. Prilikom prijave korisnika se traži unos odgovarajućeg emaila i lozinke. Funkciji "*signInWithEmailAndPassword*" nakon podnošenja podataka predaje se email i lozinka koju je korisnik prethodno upisao u za to predviđena mjesta. Nakon uspješne prijave zatvara se skočni prozor, provjerava je li korisnikov email potvrđen, te ažurira njegov status kako bi ostali korisnici znali kada je prijavljen odnosno odjavljen radi lakšeg planiranja interakcije, tj. slanja poruka.

Provjera potvrđenosti emaila ostvaruje se funkcijom "*ShowAlertMessage*" koja je prethodno objašnjena. Ažuriranje korisničkog statusa ostvarujemo funkcijom "*UpdateActiveStatus*" kojoj se predaje trenutni korisnik i na osnovu UID-a pronalazi odgovarajući čvor te unutar njega ažurira atribut. Slikom 3.12. prikazane su funkcije odgovorne za nesmetani tijek prijave korisnika. Slikom 3.13. prikazana je navedena funkcija.

```
login.addEventListener("submit", (event) => {
  event.preventDefault();
  const email = document.getElementById("login-email").value;
  const password = document.getElementById("login-password").value;

  auth.signInWithEmailAndPassword(email, password).then(function () {

    const modal = document.querySelector("#modal-login");
    M.Modal.getInstance(modal).close();
    login.reset();
    var currentUser = firebase.auth().currentUser;
    ShowAlertMessage(currentUser);
    UpdateActiveStatus(currentUser);

    location.reload();
  })
    .catch((error) => {
      window.alert(error.message);
    });
});
```

Slika 3.12. *Prikaz funkcija odgovornih za prijavljivanje korisnika*

```
function UpdateActiveStatus(currentUser) {
  db.ref("/users/" + currentUser.uid).update({
    active: true,
  });
}
```

Slika 3.13. *Prikaz funkcije UpdateActiveStatus*

Pomoću funkcije "*signOut*" korisniku je omogućeno da se odjavi sa svoga profila. Klikom na predviđeni gumb, korisnik se odjavljuje te se njegovo stanje ažurira. Ažuriranje korisničkog stanja prilikom odjave kao i prikazivanje odnosno skrivanje elemenata ovisno o korisničkom stanju, odnosno je li korisnik prijavljen ili nije, ostvaruje se funkcijom "*onAuthStateChanged*". Funkcija "*onAuthStateChanged*" funkcionira tako da prilikom svake promjene stanja, odnosno prijavljivanja i odjavljivanja, odradi određene korake. Ako je korisnik prijavljen, određeni elementi aplikacije će se prikazati, odnosno, ako je odjavljen, neki elementi će biti skriveni. Funkcija "*showElements*" je zadužena za prikazivanje i sakrivanje određenih elemenata te je prikazana slikom 3.14., dok je slikom 3.15. prikazana funkcija "*onAuthStateChanged*".

```
function showElements(user) {

  if (user) {
    loggedInList.forEach((item) => (item.style.display = "block"));
    loggedoutList.forEach((item) => (item.style.display = "none"));
    chat.style.display = "block";
    userName.style.display = "block";
    users.style.display = "block";

    if (user.emailVerified == false) {
      verification.style.display = "block";
    }
    if (user.emailVerified != false) {
      pictureDetails.style.display = "block";
      defaultPicture.getDownloadURL().then((link) => {
        const acc = `<div>
        <img src="${user.photoURL != null ? user.photoURL : link}" width="200px" style="padding-left:10px;" >
        <p style=" font-size: 16px;  border-style:  ridge ;
        border-color: #eeeeee; padding-left:10px; "> Username: ${user.displayName}</p>
        <p style="font-size: 16px;  border-style:  ridge ;
        border-color: #eeeeee; padding-left:10px; "> Email adress: ${user.email} </p>
        </div>`;

        podaci.innerHTML = acc;
      });

    }
  }
  else {
    loggedInList.forEach((item) => (item.style.display = "none"));
    loggedoutList.forEach((item) => (item.style.display = "block"));
    chat.style.display = "none";
    verification.style.display = "none";
    userName.style.display = "none";
    users.style.display = "none";
  }
};
```

Slika 3.14. *Prikaz funkcije showElements*

```
var previousUser = 1;
auth.onAuthStateChanged((user) => {
  if (user) {

    showElements(user);
    userName.innerHTML = user.displayName;
    previousUser = user;

  } else {
    showElements();
    if (previousUser != 1) {
      db.ref("/users/" + previousUser.uid).update({
        active: false,
      });
      previousUser = 1;
    }
  }
});
```

Slika 3.15. *Prikaz funkcije onAuthStateChanged*

3.2. Primanje i slanje poruka

Prijavljeni korisnik koji je potvrdio svoj email sada može koristiti sve funkcionalnosti ove aplikacije. Za prikazivanje ostalih korisnika s kojima je moguće komunicirati korišten je slušatelj (engl. *listener*) koji sluša kada će se dodati novi čvor unutar korijenskog (engl. *root*) elementa "users" u bazi podataka. Nakon što detektira da je dodan novi čvor, poziva se funkcija koja je zadužena za prikazivanje pojedinih korisnika unutar HTML elementa. Slikom 3.16. prikazan je prethodno spomenuti slušatelj.

```
userRef.on("child_added", showUsers);
```

Slika 3.16. *Prikaz slušatelja*

Funkcija koju slušatelj poziva povlači sve potrebne podatke o pojedinom korisniku. Podatke poput slike koju korisnik koristi, njegovog imena, identifikacijskog izraza te status aktivnosti. Pomoću prikupljenih podataka strukturira se izgled, odnosno reprezentacija korisnika unutar aplikacije. Prilikom svakog pozivanja, identifikacijski izraz se sprema unutar polja kako bi se omogućio pristup pojedinom korisniku pri kliku na korisničko ime. Slikom 3.17. prikazana je funkcija zadužena za obradu podataka pojedinog korisnika.

```
function showUsers(data) {
  const { slika, id, ime, active } = data.val();
  userCounter++;
  var Korisnik = `<p onclick="chooseChat(${userCounter})" style="background-color: #f5f5f5;">
  <a href="#"><img src= ${slika} width="50px" height="50px" style="border-radius: 25%;">
   <z style= "background-color:${active == true ? `green` : `red`};
    border-radius: 50%; color: ${active == true ? `green` : `red`}; font-size: 9px; margin-right:20px;"> oo</z> <br> ${ime}  </a>  </p> `;

  Korisnici.innerHTML += Korisnik;

  userID.push(id);
}
```

Slika 3.17. *Prikaz funkcije zadužene za reprezentaciju korisnika*

Kako je vidljivo na slici 3.17., prilikom klika na svakog korisnika poziva se nova funkcija pod nazivom "*chooseChat*" kojoj je predana vrijednost brojača. Navedena funkcija prilikom klika na određenog korisnika postavlja ime te osobe na vrh razgovornog prozora kako bi bilo jasnije kome su upućene poruke. Funkcija također uklanja određene slušatelje kako ne bi došlo do grešaka i ponavljanja u učitavanju poruka te postavlja slušatelja kako bi se prikazale razgovorne poruke između dva specifična korisnika. Prethodno spomenuti slušatelj poziva funkciju koja je zadužena za manipuliranje, odnosno prikazivanje, poruka na određeni način prilikom svake novododane poruke unutar baze podataka. Slikom 3.18. prikazana je funkcija "*chooseChat*".

```
function chooseChat(userCounter) {

  messageScreen.innerHTML = "";
  unreadMessages.innerHTML = "";
  numberOfUnreadMessages.innerHTML = "";
  unreadMessagesCounter = 0;

  msgRef.off("child_added", checkMessages);
  msgBtn.removeAttribute("disabled");

  chosenNameID = userID[userCounter - 1];

  var chosenName = db.ref("/users/" + userID[userCounter - 1] + "/ime");

  chosenName.on("value", function (snapshot) {
    chatName.innerHTML = snapshot.val();
  });

  msgRef.on("child_added", checkMessages);
}
```

Slika 3.18. *Prikaz funkcije choseChat*

Nakon što je izabran korisnik s kojim se želi komunicirati potrebno je unijeti poruku unutar input elementa te nakon toga pritisnuti tipku *Send* (engl. *button*). Na tipku *Send* postavljen je slušatelj te prilikom klika se provodi funkcija koja je zadužena za provjeru duljine unesene poruke. Ukoliko je poruka prazan prostor, odnosno ukoliko ništa nije uneseno, funkcija neće ništa unijeti u bazu. Ukoliko je poruka duža od 75 znakova, ostatak će biti prelomljen u novi redak kako ne bi došlo do poremećaja u dizajnu te se poziva funkcija "*AddMessageToDatabase*" koja je zadužena za unos poruke unutar baze podataka. Slikom 3.19. prikazana je funkcija koja je zadužena za provjeru duljine unesene poruke.

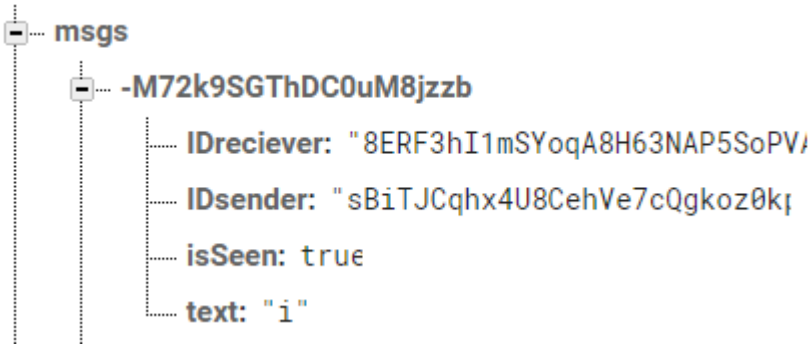
```
msgBtn.addEventListener("click", (event) => {
  event.preventDefault();

  if (!msgInput.value.trim()) {
    return;
  }
  var message = msgInput.value;
  msgInput.value = "";
  for (var n = 0; msgInput.value.trim().length >= n;) {
    var firstPart = message.slice(n, n + 75);
    var secondPart = message.slice(n, n + 76);

    if (firstPart.slice(74) != " " && secondPart.slice(75) != " " && firstPart.length == 75) {
      msgInput.value += firstPart + "-" + "<br>";
    }
    else if (firstPart.length == 75) {
      msgInput.value += firstPart + "<br>";
    }
    else {
      msgInput.value += firstPart;
    }
    n = n + 75;
  }

  AddMessageToDatabase(msgInput)
});
```

Slika 3.19. *Prikaz funkcije zadužene za provjeru duljine poruke*

Funkciji "*AddMessageToDatabase*" se predaje unesena poruka te funkcija strukturira i unosi poruku zajedno s ostalim informacijama u bazu podataka. Atribut pod nazivom "*isSeen*" postavlja se na vrijednost false, unosi se jedinstveni identifikacijski niz pošiljatelja, primatelja te na kraju i sama poruka. Tako strukturirani izraz unosi se u bazu podataka pomoću funkcije "*push*" koja sama generira nasumični izraz korijenskog (engl. *root*) elementa. Slikom 3.20. prikazana je funkcija zadužena za strukturiranje i unošenje poruka u bazu podataka. Slikom 3.21. prikazana je poruka unutar baze podataka.

```
function AddMessageToDatabase(msgInput) {
  var currentUser = firebase.auth().currentUser;

  const msg = {
    isSeen: false,
    IDsender: currentUser.uid,
    IDreciever: chosenNameID,
    text: msgInput.value,
  };
  msgRef.push(msg);
  msgInput.value = "";
}
```

Slika 3.20. *Prikaz funkcije AddMessageToDatabase*



Slika 3.21. *Prikaz strukture poruke unutar baze podataka*

Nakon što je poruka uspješno spremljena u bazu podataka prethodno spomenuti slušatelj (engl. *listener*) poziva funkciju pod nazivom "*checkMessages*" kojoj se predaje cijeli strukturirani čvor pojedine poruke. Ova funkcija poziva nekoliko drugih funkcija kao što je funkcija za ažuriranje obavijesti nepročitanih poruka, funkcija koja ažurira razgovorni prozor, funkcija za ažuriranje atributa *isSeen* unutar baze podataka i slično. Slikom 3.22. prikazana je funkcija "*checkMessages*".

```
function checkMessages(data) {

  seenMsg.style.display = "none ";
  updateUnreadMessages(data);
  updateMessageScreen(data);
  updateSeen(data);
  messageScreen.scrollIntoView(false);
  ShowSeen(data);
}
```

Slika 3.22. *Prikaz funkcije checkMessages*

Funkciji "*updateUnreadMessages*" predaju se sve informacije pojedine poruke te ažurira i strukturira prikaz pojedine poruke koja nije pročitana. Provjerava se je li trenutno prijavljeni korisnik primatelj poruke te se provjerava je li trenutno odabrana osoba pošiljatelj. U tom slučaju, poruka je pogledana i nema potrebe za obavještavanjem o nepročitanim porukama. Obavijesti o nepročitanim porukama prikazane su tako da se prikaže slika profila pošiljatelja, njegovo ime te kako poruka glasi. Slikom 3.23. prikazana je spomenuta funkcija.

```
function updateUnreadMessages(data) {

  const { IDsender, IDreciever, text, isSeen } = data.val();
  var senderName = db.ref("/users/" + IDsender + "/ime");
  var name;
  var currentUser = firebase.auth().currentUser;

  senderName.on("value", function (snapshot) {
    name = snapshot.val();

    if (currentUser.uid == IDreciever && isSeen == false && chosenNameID != IDsender) {
      unreadMessagesCounter++;
      var picture = db.ref("/users/" + IDsender + "/slika");
      picture.on("value", (snapshot) => {
        profilePic = snapshot.val();

        unreadMessages.innerHTML += ` <div style="  border-style:  ridge ; border-color: #eeeeee;">
        <img src="${profilePic}" width=50px height=50px />  ${name}:  ${text} </div> `;
      });
    }
    numberOfUnreadMessages.innerHTML = `Unread messages (${unreadMessagesCounter})`;
  });
}
```

Slika 3.23. *Prikaz funkcije updateUnreadMessages*

Funkciji "*updateMessageScreen*" predaju se informacije o pojedinoj poslanoj poruci. Ova funkcija povlači informacije o pošiljatelju, primatelju i samom sadržaju poruke te ažurira razgovorni prozor. Ako je trenutni korisnik primatelj poslane poruke i ako je pošiljatelj korisnik koji je prethodno odabran, odnosno kliknut, ili, ako je odabrani korisnik primatelj i trenutno prijavljeni

korisnik pošiljatelj, poruke će se prikazati na određeni način. Slikom 3.24. prikazana je spomenuta funkciju.

```
function updateMessageScreen(data) {

  const { IDsender, IDreciever, text } = data.val();
  var currentUser = firebase.auth().currentUser;

  if ((IDreciever == currentUser.uid && IDsender == chosenNameID) || (IDreciever == chosenNameID && IDsender == currentUser.uid)) {
    msg = `<li class=${IDsender == currentUser.uid ? `msg-my` : `msg`}> <span>  ${text}</span> </li>`;
    messageScreen.innerHTML += msg;
  }

}
```

Slika 3.24. *Prikaz funkcije updateMessageScreen*

Funkciji "*updateSeen*" predaju se informacije o pojedinoj poslanoj poruci. Ova funkcija povlači informacije o pošiljatelju, primatelju i o tome je li poruka pogledana ili nije. Prvo je potrebno provjeriti je li poruka prethodno pročitana te identitet pošiljatelja, odnosno primatelja, i naravno, stanje aktivnosti korisnika. Funkcija također poziva drugu funkciju koja je zadužena za prikazivanje kratke poruke *Seen* ukoliko je poruka pročitana kako bi se dalo do znanja pošiljatelju da je primatelj pogledao poruku. Slikom 3.25. prikazana je funkcija koja je zadužena za ažuriranje i provjeru određenog podatka unutar baze podataka.

```
function updateSeen(data) {

  const { IDsender, IDreciever, isSeen } = data.val();
  var currentUser = firebase.auth().currentUser;
  var ActiveStatus;
  var Active = db.ref("/users/" + IDreciever + "/active");

  Active.on("value", snapshot => {
    ActiveStatus = snapshot.val();
  })

  if (chosenNameID == IDsender && ActiveStatus == true && currentUser.uid == IDreciever && isSeen == false) {
    db.ref("/msgs/" + data.key).update({
      isSeen: true,
    })
  }

  if (currentUser.uid == IDsender && chosenNameID == IDreciever && isSeen == false) {
    msgRef.once("child_changed", ShowSeen);
  }
}
```

Slika 3.25. *Prikaz funkcije updateSeen*

Funkciji "*showSeen*" predaju se informacije o pojedinoj poslanoj poruci. Funkcija se poziva prilikom svake poruke unutar funkcije "*checkMessages*" i dodatno se poziva ako je vrijednost atributa *isSeen* false zbog slučaja kada je trenutno otvoren razgovorni prozor, jer poruka *Seen* se ne bi prikazivala zato što je potrebno neko vrijeme kako bi se ažurirala informacija unutar baze

podataka te se upravo zato dodatno poziva unutar funkcije "*updateSeen*". Slikom 3.26. prikazana je funkcija zadužena za prikazivanje kratke poruke *Seen*.

```
function ShowSeen(data) {
  const { IDsender, IDreciever, isSeen } = data.val();
  var currentUser = firebase.auth().currentUser;

  if (isSeen == true && chosenNameID == IDreciever && currentUser.uid == IDsender) {
    seenMsg.style.display = "block";
    seenMsg.scrollIntoView();
  }
}
```

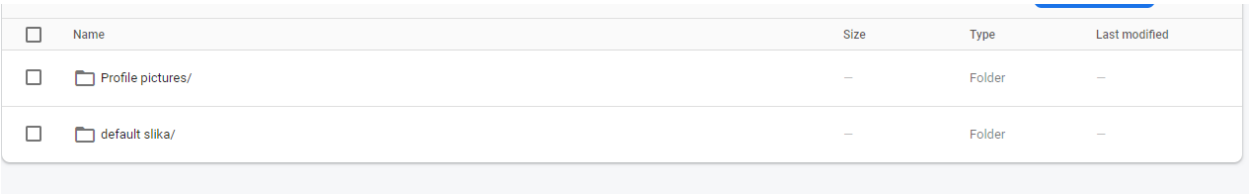
Slika 3.26. *Prikaz funkcije ShowSeen*

3.3. Ostale funkcionalnosti

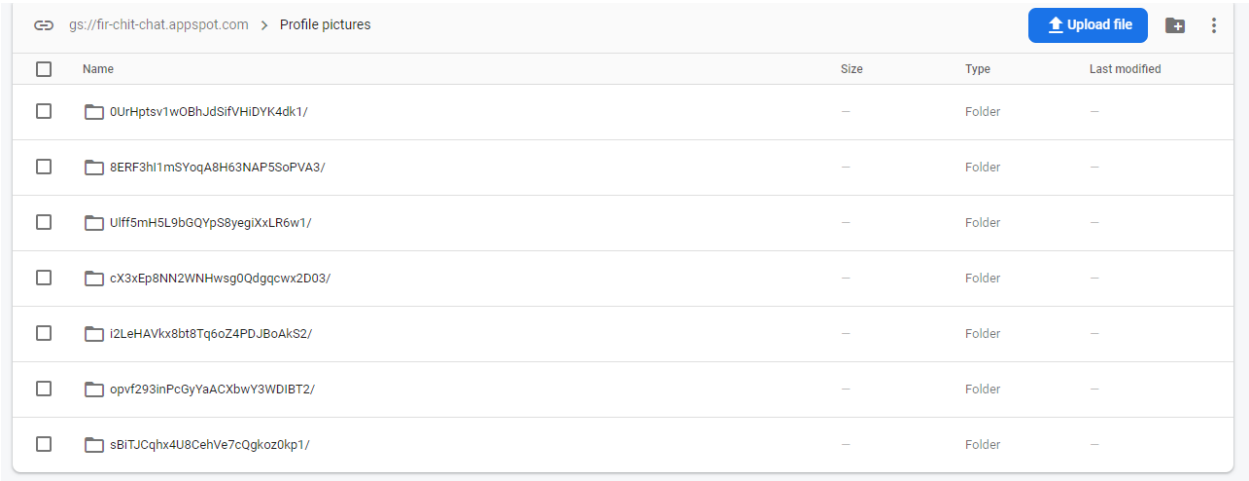
Kako bi korisnici mogli biti prepoznatljivi i jedinstveni unutar aplikacije dodana je mogućnost promjene slike profila. Prilikom klika na tipku koja se naziva jednako kao i sam korisnik, pojavljuje se skočni prozor s podatcima o korisniku, ime, trenutna slika, email te mogućnost učitavanja (engl. *upload*) nove profilne slike koja će biti vidljiva svim korisnicima. Nakon što se odabere željena slika, korisnika se obavještava kako je potrebno pričekati nekoliko trenutaka kako bi se slika spremila unutar Firebase Storage te nakon toga preuzela te prikazala kao profilna slika. Slikom 3.27. prikazana je funkcija zadužena za promjenu profilne slike. Slikom 3.28. i 3.29. prikazana je struktura pohranjenih slika unutar Firebase Storage.

```
photo.addEventListener("change", (e) => {
  var currentUser = firebase.auth().currentUser;
  const file = e.target.files[0];
  const name = file.name;
  const metadata = {
    contentType: file.type,
  };
  const Storageref = storageDb.ref("Profile pictures/" + currentUser.uid);
  alert("Pričekajte nekoliko trenutaka učitavanje slike profila.");

  Storageref.child(name).put(file, metadata).then((snapshot) => snapshot.ref.getDownloadURL()).then((link) => {
    currentUser.updateProfile({ photoURL: link }).then(function () {
      db.ref("/users/" + currentUser.uid).update({
        slika: link,
      });
      location.reload();
    }).catch(function (error) { window.alert(error.message) });

  });
});
```

Slika 3.27. *Prikaz funkcija zaduženih za ažuriranje slike profila*



Slika 3.28. *Prikaz datoteka unutar Firebase Storage*



Slika 3.29. *Prikaz načina pohrane slika profila pojedinih korisnika*

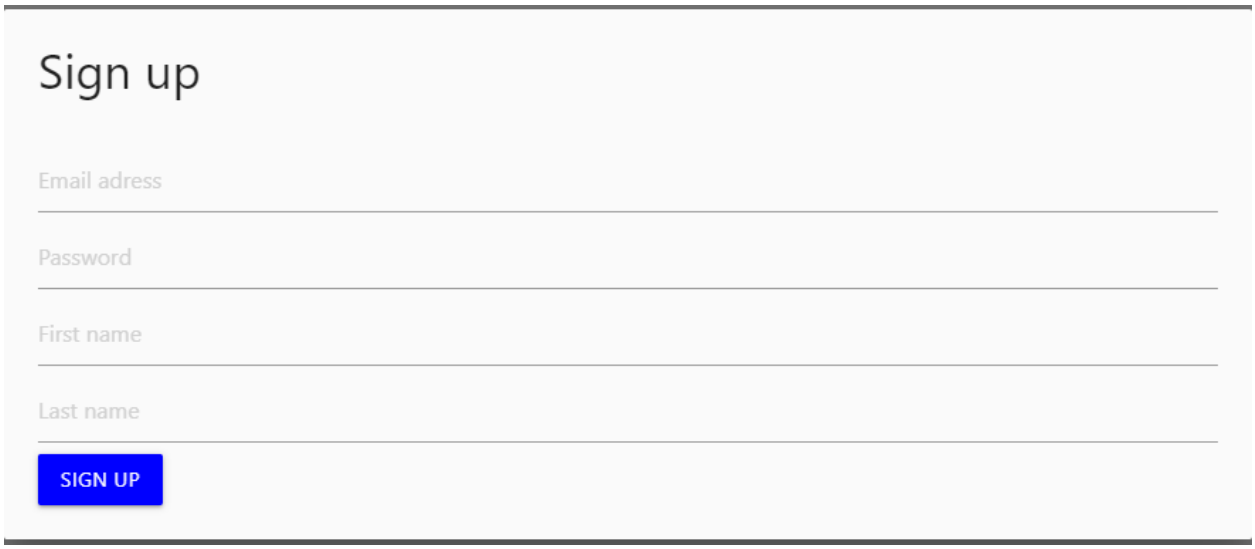
Korisnike je osim prema slici profila moguće pretražiti i po imenu. Ukoliko ima više korisnika s istim imenom, svi korisnici će biti prikazani. Navedena funkcionalnost postiže se pretraživanjem baze podataka. Nakon što korisnik unese određeno ime i klikne na tipku *Search*, pretraga se vrši na način da se traže svi korisnici kojima je ime jednako imenu unutar tražilice. Slikom 3.30. prikazana je navedena funkcija.

```
function Search() {
  const SearchName = document.getElementById("searchName").value;
  Users.innerHTML = "";
  userRef.orderByChild("ime").equalTo(SearchName).on("child_added", showUsers);
}
```

Slika 3.30. *Prikaz funkcije Search*

4. VIZUALNA REPREZENTACIJA APLIKACIJE

U nastavku je prikazan i ukratko prokomentiran izgled završene aplikacije koja sadržava sve funkcionalnosti koje su prethodno navedene i ukratko objašnjene. Slikom 4.1. prikazan je izgled forme za registraciju. Registracijska forma sastoji se od nekoliko HTML <input> oznaka koje služe za unošenje email adrese, zaporke, imena i prezimena te se na dnu forme nalazi *Sign up* tipka. Prilikom registracije potrebno je unijeti email, zaporku, ime i prezime.



Slika 4.1. *Prikaz forme za registraciju*

Slikom 4.2. prikazana je forma za prijavu (engl. *login*) kod koje je potrebno unijeti odgovarajući email i lozinku. Forma za prijavu sastoji se od dvije HTML <input> oznake koje služe za unošenje odgovarajućih informacija odnosno emaila i zaporke te ukoliko se jedno od navedenih polja ostavi prazno ili se ne unesu odgovarajući podatci, prijava nije moguća.

Slika 4.2. *Prikaz forme za prijavu*

Slika 4.3. *Prikaz gotove aplikacije*

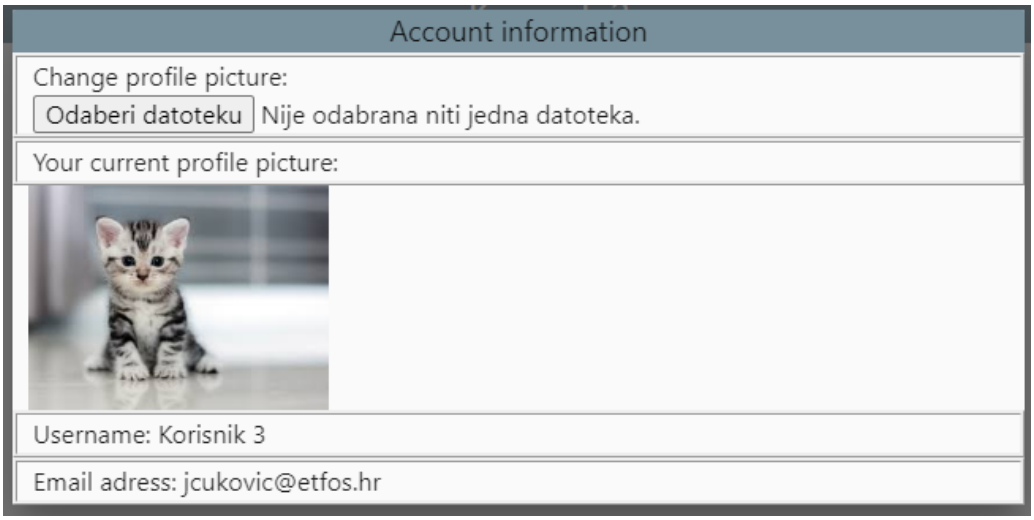
Na lijevoj strani slike 4.3. prikazani su svi korisnici te njihova imena i slike profila. Zelenim, odnosno crvenim, krugom kraj slike profila prikazano je stanje korisnika odnosno jesu li prijavljeni ili odjavljeni. Iznad liste korisnika nalazi se tražilica kojom se po imenu pretražuju korisnici, a sve što je potrebno je unijeti ime željenog korisnika i pritisnuti tipku *Search*. U sredini se nalazi razgovorni prozor koji se sastoji od nekoliko dijelova. Na samom vrhu nalazi se ime korisnika s kojim se razmjenjuju poruke, u ovom slučaju to je korisnik pod imenom "Korisnik 2". Vidljive su dvije poruke poslane između korisnika pod imenima "Korisnik 2" i "Korisnik 3" te poruka *Seen* koja označava da je "Korisnik 2" pročitao poruku. Na dnu samog razgovornog prozora nalazi se element za unos poruka, a njemu s desna tipka *Send* s kojom se šalje poruka koja je unesena u za to predviđeno mjesto. Iznad razgovornog prozora nalazi se tipka za odjavu, ime prijavljenog korisnika te broj nepročitanih poruka. Slikama 4.4., 4.5. prikazan je HTML kod svih navedenih elemenata aplikacije. Prilikom klika na ime prijavljenog korisnika skočni prozor prikazan slikom 4.6. postaje vidljiv, a skočnim prozorom prikazani su podatci o korisniku te je ponuđena mogućnost promjene slike profila.

```html
<div id="modal-messages" class="modal">
  <p id="unreadMessagesHeader">Unread messages</p>
  <div id="unreadMessages"></div>
</div>

<div class="row">
  <div class="Users col s4 m2 l2 logged-in" id="Users"></div>
  <div class="chat col s8 offset-s6 m9 offset-m5 l9 offset-l4 logged-in">
    <div class="row">
      <div class="chat-header col s12" id="chat-header"> Chit-chat</div>

      <div class="chat-window col s12">

        <ul id="messages"></ul>
        <p id="SeenMsg"> Seen</p>

        <form id="msgform">
          <input type="text" class="col s6 m7 l8" id="msg-input" placeholder="Enter a message" autocomplete="off" />
          <button id="msg-btn" disabled>Send</button>
        </form>

      </div>
    </div>
  </div>

</div>
```

Slika 4.4. *Prikaz HTML koda razgovornog prozora i svih njegovih elemenata, korisnika te kod nepročitanih poruka*

```html
<ul>
  <li class="logged-in" id="SearchPeople">
    <input type="text" id="searchName" placeholder="Search">
  </li>
  <li class="logged-in">
    <button onclick="Search()" id="SearchBtn">Search</button>
  </li>
</ul>
```

Slika 4.5. *Prikaz HTML koda elemenata za pretragu korisnika po imenu*

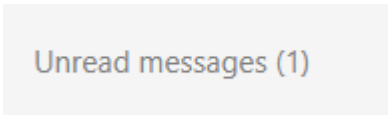


Slika 4.6. *Prikaz informacija korisnika*

```
<div id="modal-account" class="modal">
  <p id="accountDetailsHeader">Account information</p>
  <div class="pictureDetails" id="pictureDetails">
    <p>Change profile picture: </p>
    <p> <input type="file" value="upload" id="photo" /> </p>
  </div>
  <div class="currentPicture">
    <p class="accountDetails"> Your current profile picture:</p>
  </div>
  <div id="account-details" class="accountDetails"></div>
</div>
```

Slika 4.7. *HTML kod modula za prikazivanje informacija korisnika*

Slikom 4.8. prikazan je broj nepročitanih poruka. Nakon što kliknemo na nepročitane poruke (engl. *unread messages*) prikazuje se skočni prozor koji sadrži sliku profila korisnika koji šalje poruku, njegovo ime te samu poruku. Slikom 4.9. prikazan je način prikazivanja nepročitanih poruka.



Slika 4.8. *Prikaz broja nepročitanih poruka*

Slika 4.9. *Prikaz nepročitanih poruka*

5. ZAKLJUČAK

Među ljudima se javlja sve veća potreba za komuniciranjem putem interneta, kako u poslovnom, tako i u privatnom životu, a s pojavom iste raste i potreba za razvojem aplikacija koje će im to omogućiti. U ovom završnom radu prikazan je postupak izrade web aplikacije za chat koja pruža ispunjenje prethodno navedenih zahtjeva današnjice.

Tijekom izrade korištene su razne tehnologije koje su omogućile uspješnu realizaciju sustava registracije i prijave korisnika kao i razmjenu samih poruka. Aplikacija omogućava različitim korisnicima međusobno povezivanje i komunikaciju, funkcionalnosti kao što su personalizacija korisnika promjenom slike profila, pretraga korisnika po imenu te uvid u poruke koje nisu pročitane, što je ujedno bio cilj, pa se može zaključiti kako je uspješno ispunjen zadatak završnog rada.

Velika prednost izrade ovakve web aplikacije je samostalni napredak i mogućnost učenja, ali i usvajanje znanja različitih tehnologija korištenih u web programiranju kao što su HTML, CSS, JavaScript i Firebase. Navedene tehnologije uvelike su zastupljene općenito prilikom izrade web stranica, a dobro poznavanje istih povećava prepoznatljivost i konkurentnost programera.

LITERATURA

[1] W3Schools, HTML Tutorial, https://www.w3schools.com/html/html_intro.asp, pristupljeno 17.Svibnja.2020.

[2] W3Schools, CSS Tutorial, https://www.w3schools.com/css/default.asp,

pristupljeno 17.Svibnja.2020.

[3] W3Schools, How To Add CSS, https://www.w3schools.com/css/css_howto.asp, pristupljeno 17. svibnja 2020.

[4] Materialize, About, https://materializecss.com/about.html,

pristupljeno 22. svibnja 2020.

[5] Materialize, Getting Started, https://materializecss.com/getting-started.html,

pristupljeno 22. svibnja 2020.

[6] Wikipedia, JavaScript, https://hr.wikipedia.org/wiki/JavaScript,

pristupljeno 22. svibnja 2020.

[7] Wikipedia, JavaScript, https://en.wikipedia.org/wiki/JavaScript,

pristupljeno 22. svibnja 2020.

[8] Wikipedia, Firebase, https://en.wikipedia.org/wiki/Firebase,

pristupljeno 25. svibnja 2020.

SAŽETAK

Temeljni zadatak ovog završnog rada bio je izraditi web aplikaciju koja omogućava komunikaciju između korisnika u stvarnom vremenu. HTML, CSS, Firebase, JavaScript i Materialize u radu su teorijski objašnjeni i primijenjeni tijekom izrade aplikacije. U ovome radu je prikazana izvedba svih funkcionalnosti i mogućnosti ove web aplikacije za komunikaciju, odnosno za chat. Za spremanje svih informacija o porukama i korisnicima u bazu podataka te za omogućavanje prijave i registracije korisnika koristio se Firebase.

Ključne riječi: baza podataka, funkcija, komunikacija, korisnik, poruka

ABSTRACT

Title: Web chat application

Fundamental task of this paper was to create a web application which provides real-time communication between users. HTML, CSS, Firebase, JavaScript and Materialize are theoretically explained in this paper and applied in the making of the application. This paper presents the implementation of all functionalities and possibilities of this web application for communication, ie for chat. For storing all information about messages and users in the database and enabling user login and registration was used Firebase.

Key words: database, function, communication, user, message

ŽIVOTOPIS

Josip Čuković rođen je 3. srpnja 1998. godine u Đakovu, Hrvatska. Osnovnoškolsko te srednjoškolsko obrazovanje završava u Đakovu. Srednju školu "Srednja strukovna škola Antuna Horvata, Đakovo" smjer "Računalni tehničar za strojarstvo" završava 2017. godine. Nakon završetka srednjoškolskog obrazovanja i uspješnog polaganja državne mature, upisuje se na preddiplomski studij računarstva na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku.

PRILOZI

CD:

1. Završni rad „Web aplikacija za chat.docx“
2. Završni rad „Web aplikacija za chat.pdf“
3. Izvorni kod web aplikacije